AMR in Titanium

Tong Wen and Phil Colella
ANAG, LBNL

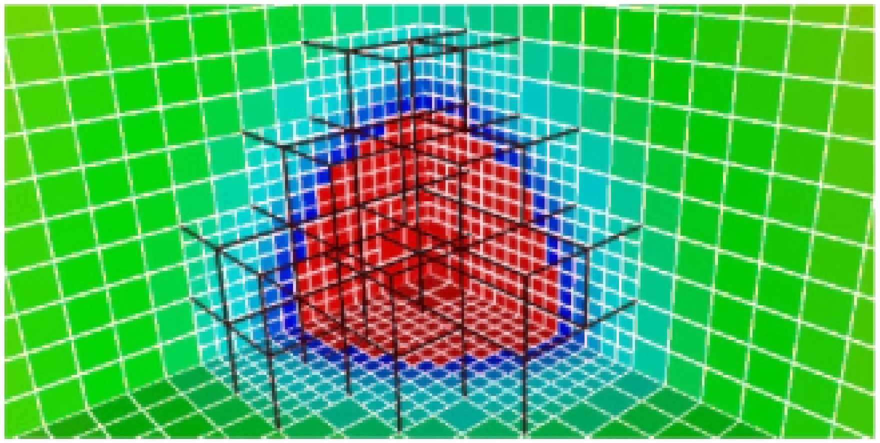
U.C. Berkeley
September 9, 2004

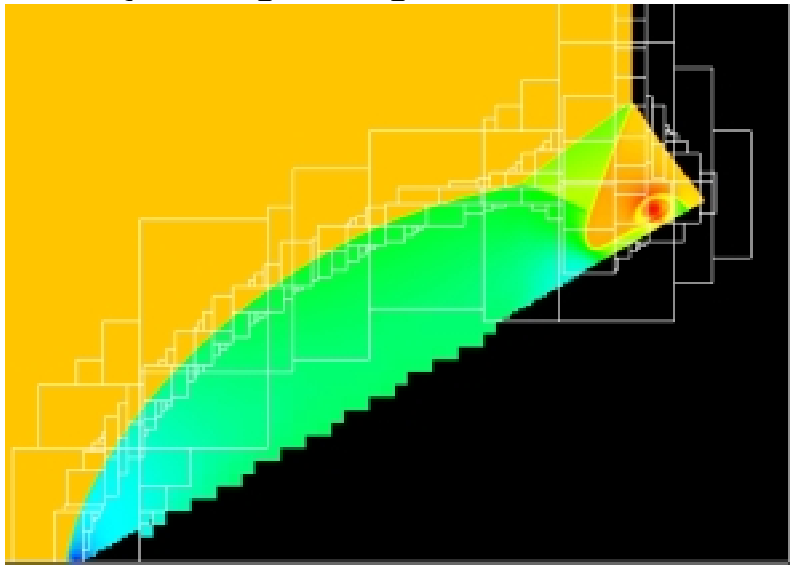
Overview

- **Our goal:**
  1. First, build the infrastructure for AMR applications in Titanium.
  2. Meanwhile, provide a test case for Titanium's performance and programmability.
  3. Finally, make it easier to develop new AMR algorithms in this environment.
- **Content:**
  1. Block-structured adaptive mesh refinement(AMR).
  2. Titanium AMR.
  3. The test problems and profiling results.
  4. Conclusion and future work.

Local Refinement for Partial Differential Equations

- **A variety of problems exhibit multiscale behavior, in the form of localized large gradients separated by large regions where the solution is smooth.**



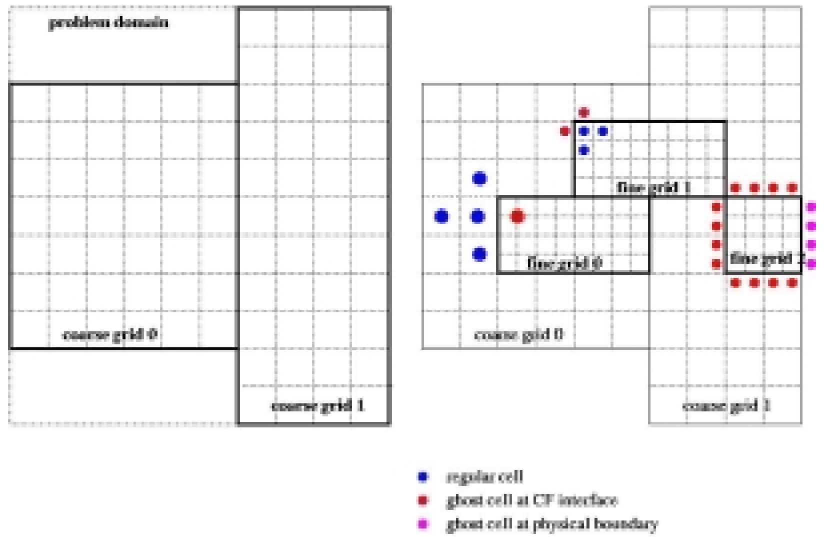


- **In adaptive methods, one adjusts the computational effort locally to maintain a uniform level of accuracy throughout the problem domain.**

Why is Block-Structured AMR Difficult?

Simplicity is traded for computational resources in AMR.

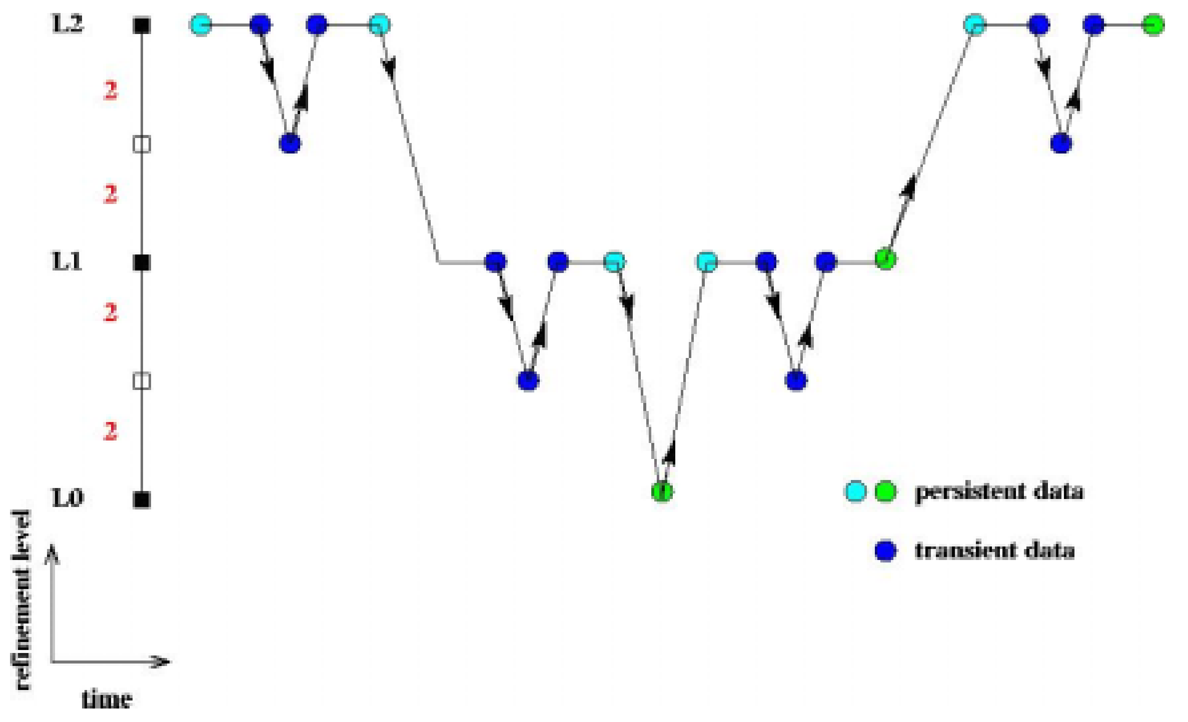
- Mixture of regular and irregular data access and computation.



1. Copy boundary values from adjacent grids at the same refinement level(irregular communication).
2. Interpolate boundary values from coarse/fine grids(irregular communication and computation).
3. evaluate finite difference on each grid(regular computation).

Why is Block-Structured AMR Difficult?

- **Complicated control structures and interactions between levels of refinement.**

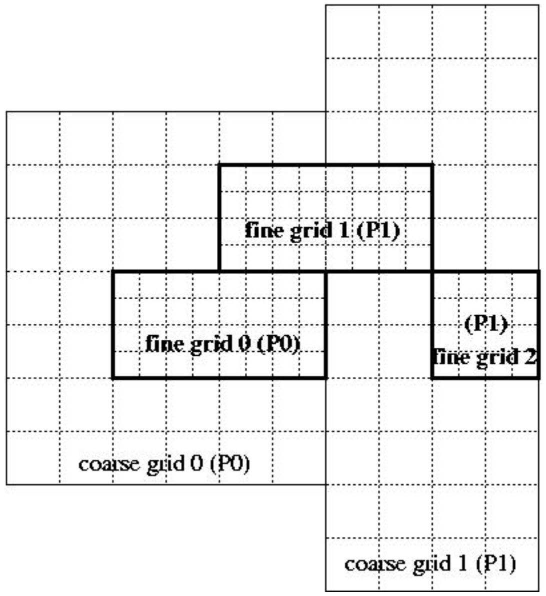


Titanium Chombo

- **Prior experience:**
  1. Early Fortran77 implementation.
  2. C++/Fortran hybrid(BoxLib, Chombo):
     - complicated data structures and irregular computations in C++.
     - Fortran to evaluate operations on rectangular arrays.
- **Current approach:**
  - Follow the Chombo design.
  - Bulk-synchronous communication:
    1. communicate boundary data for all grids at a level.
    2. perform local calculation on each grid in parallel.

Basic AMR Data Structures Build on Top of Titanium

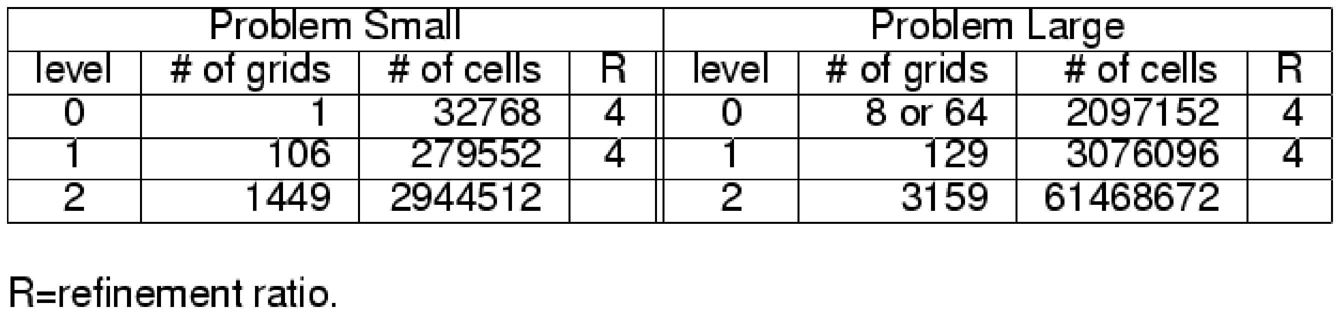
- **BoxTools: Data and operations on unions of RectDomains(grids, boxes).**
  - The metadata class: an array of RectDomains at the same refinement level along with their processor assignments.



  - The data class: defined on the metadata class, an array of distributed objects defined on the RectDomains contained in the metadata class. Each object resides on the processor its RectDomain is assigned to.

Two Test Problems

- **Solving Poisson equation with two grid configurations(3-D Vortex Ring Problem).**

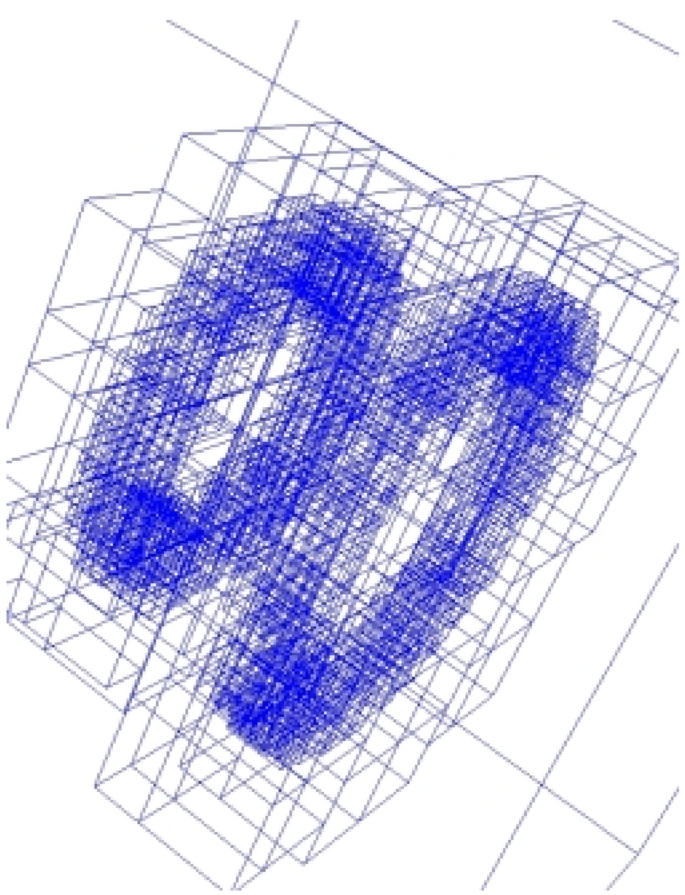


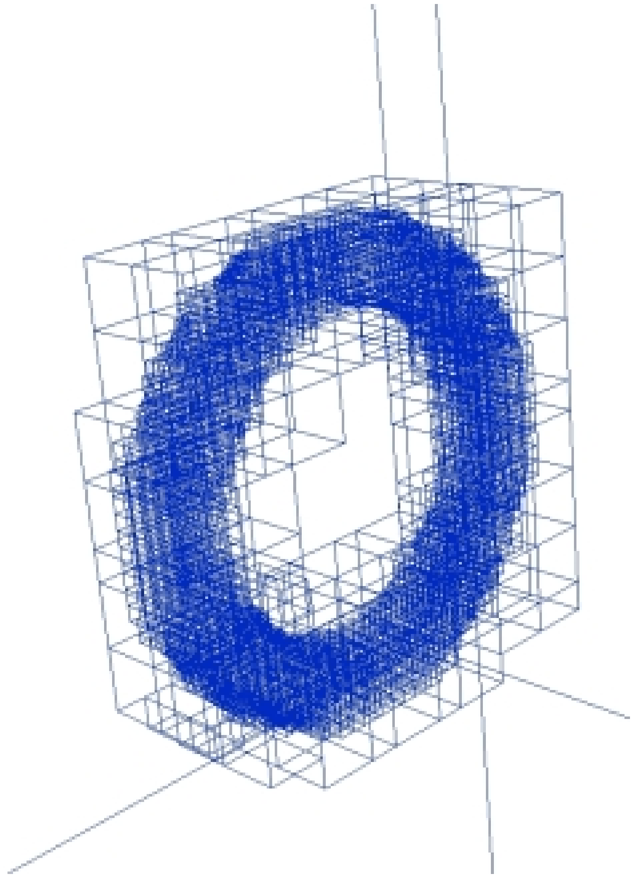
Problem Small				Problem Large			
level	# of grids	# of cells	R	level	# of grids	# of cells	R
0	1	32768	4	0	8 or 64	2097152	4
1	106	279552	4	1	129	3076096	4
2	1449	2944512		2	3159	61468672	

R=refinement ratio.

  - Can be many grids at each level.
  - In real applications, grid configuration is not known until runtime, and changes at runtime.

Grid Configurations

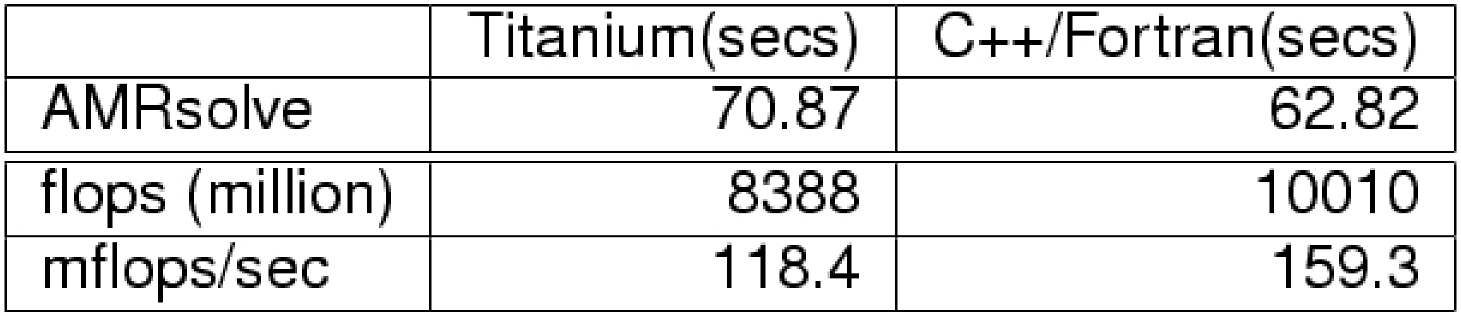




Each box represents a grid and it contains several thousands cells.

Serial Performance

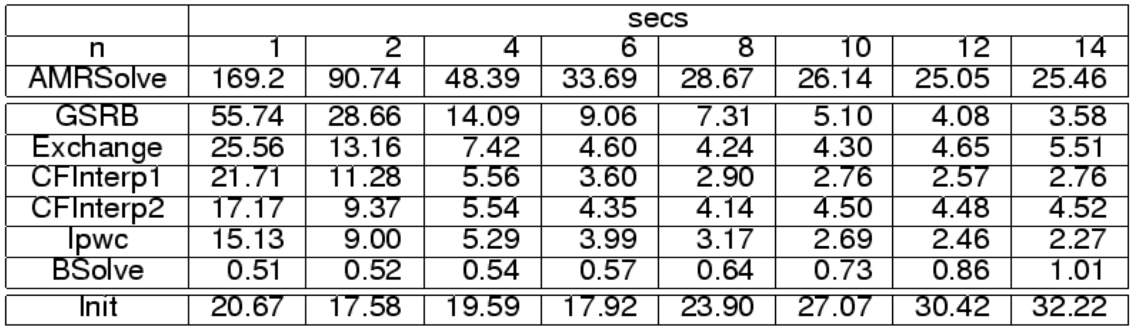
- **On two platforms(IBM SP and Pentium III workstation), the performance of our Poisson solver on the small problem matches that of Chombo.**
- **On Seberg.lbl.gov(Pentium III workstation), titanium-2.279:**



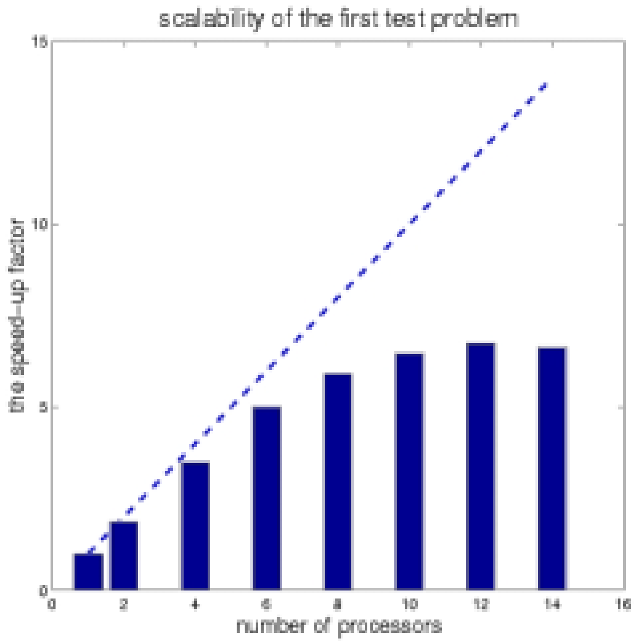
	Titanium(secs)	C++/Fortran(secs)
AMRsolve	70.87	62.82
flops (million)	8388	10010
mflops/sec	118.4	159.3

Scalability of the Small Problem

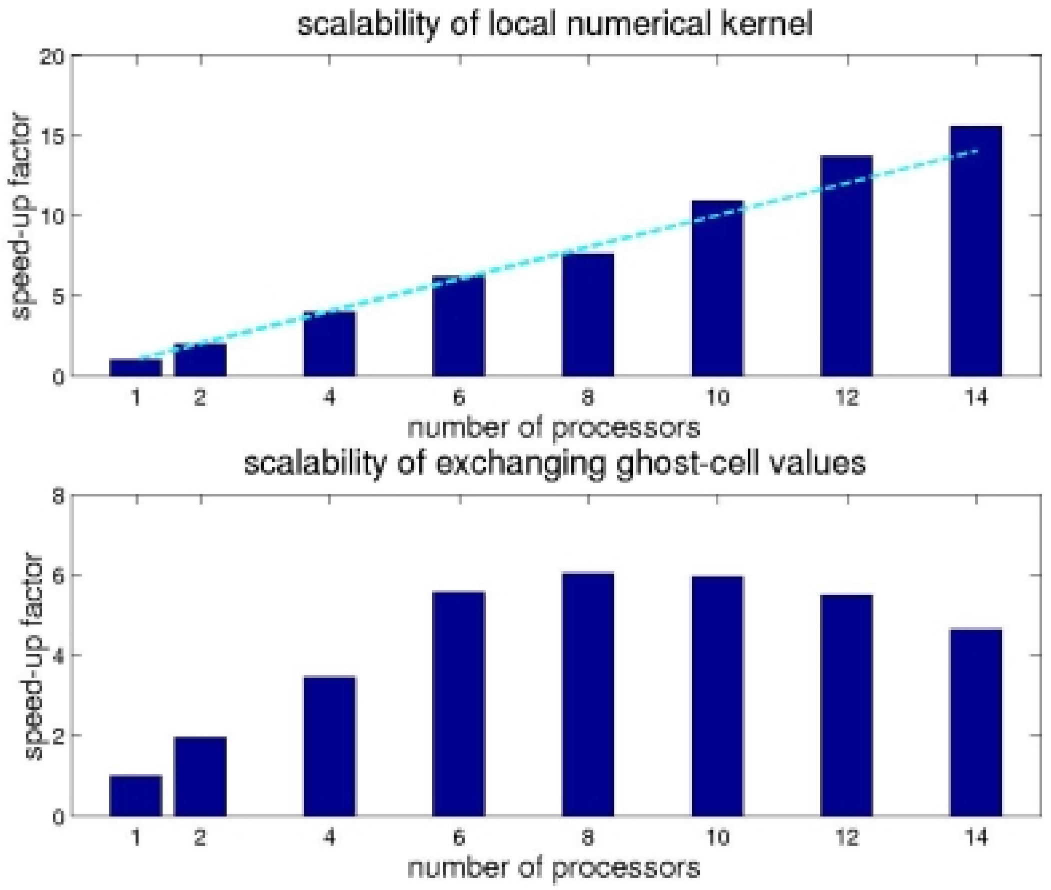
- **On Seaborg(IBM SP), titanium-2.573:**



	secs							
n	1	2	4	6	8	10	12	14
AMRSolve	169.2	90.74	48.39	33.69	28.67	26.14	25.05	25.46
GSRB	55.74	28.66	14.09	9.06	7.31	5.10	4.08	3.58
Exchange	25.56	13.16	7.42	4.60	4.24	4.30	4.65	5.51
CFInterp1	21.71	11.28	5.56	3.60	2.90	2.76	2.57	2.76
CFInterp2	17.17	9.37	5.54	4.35	4.14	4.50	4.48	4.52
Ipwc	15.13	9.00	5.29	3.99	3.17	2.69	2.46	2.27
BSolve	0.51	0.52	0.54	0.57	0.64	0.73	0.86	1.01
Init	20.67	17.58	19.59	17.92	23.90	27.07	30.42	32.22



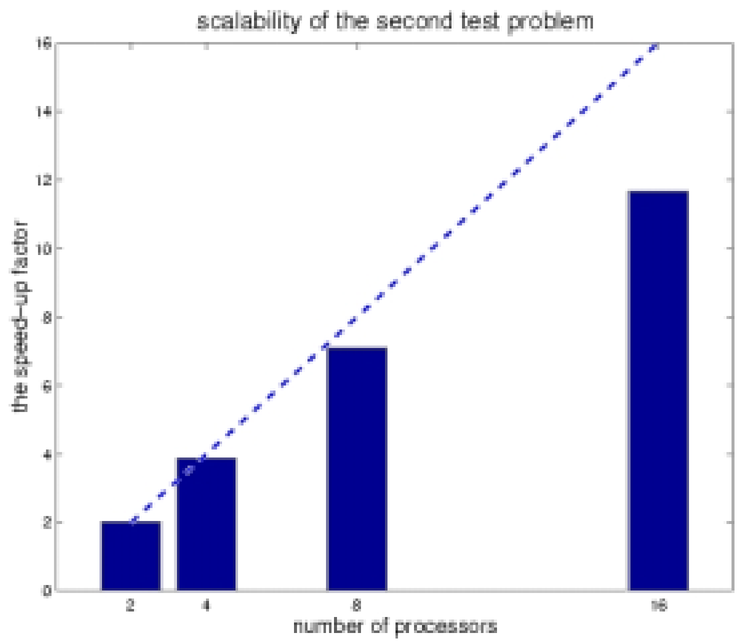
Scalability of Titanium AMR



Scalability of the Large Problem

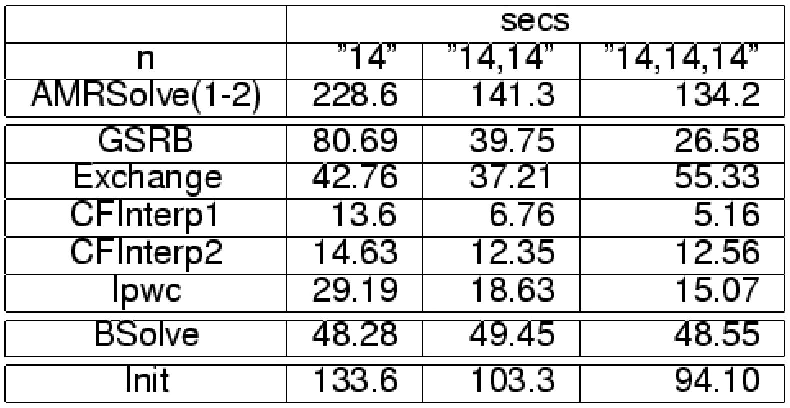
- **On Seaborg(IBM SP), titanium-2.573, 64bit:**

	secs			
n	2	4	8	16
AMRSolve(1-2)	1238	639.0	348.7	212.2
GSRB	504.1	257.5	133.5	72.1
Exchange	127.2	86.89	56.46	41.2
CFInterp1	65.58	34.74	19.34	13.16
CFInterp2	65.57	34.91	22.6	12.67
lpwc	183.0	92.66	48.46	26.15
BSolve	48.37	47.95	47.97	48.61
Init	101.7	104.5	109.7	134.5

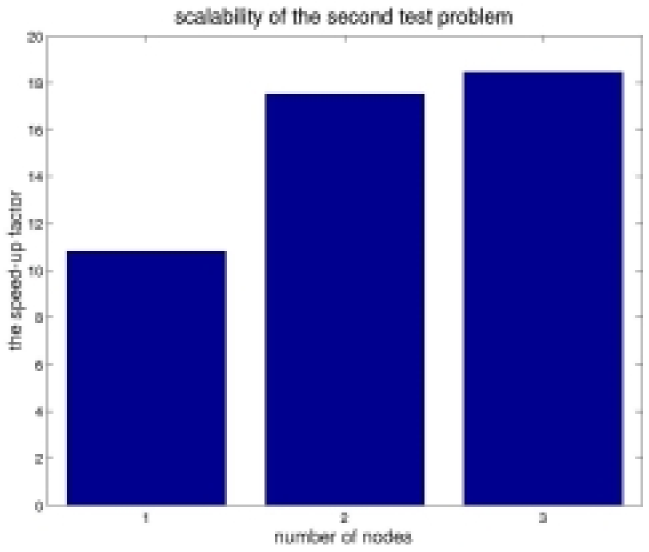


Scalability of the Large Problem

- **On Seaborg(IBM SP), titanium-2.573, 64bit:**



	secs		
n	"14"	"14,14"	"14,14,14"
AMRSolve(1-2)	228.6	141.3	134.2
GSRB	80.69	39.75	26.58
Exchange	42.76	37.21	55.33
CFInterp1	13.6	6.76	5.16
CFInterp2	14.63	12.35	12.56
Ipwc	29.19	18.63	15.07
BSolve	48.28	49.45	48.55
Init	133.6	103.3	94.10



- **A speed-up factor 20 is achieved(the goal is 30-35).**

Conclusion and Future Work

- **Titanium’s strength: language-level, one-sided high-performance communication.**
- **Major improvements of Titanium motivated by this project:**
  1. The new domain library.
  2. Fully supported template functionality.
- **Future work:**
  - Improve the performance of AMR exchange.
  - New AMR development: ocean modeling.
    - Poisson solver for problems with thin layers(testing).